# アルハンブラタイルのグラフィックスー幾何と作図 (J8 版)

SHIMURA Masato
JCD02773@nifty.ne.jp

2017 年 12 月 8 日

## 目次



アルハンブラタイルを描くのに使った作図法と幾何をまとめておく。

## グラナダ/イスラムの黄金時代

- 4-5 世紀にキリスト教徒がアレキサンドリアの図書館を破壊して、ギリシャ文明は途絶えた。
- 石油がまだ無いころにアラブの黄金時代があった。
- イベリア半島では
  - アルハンブラっ宮殿の建造された頃はヨーロッパはまだ暗黒時代、イスラム文化の最盛期であった。
  - カトリックに改宗した西ゴートがイスラムに征服され (711)、谷間に逃げ込んだ小王国からレコンキスタが始まって、セビーリアにまで及んだが、それから 250 年グラナダは持ちこたえ、イスラム王朝が栄えた。
  - ムスリムは支配地域をムスリムへの改宗を強制せず、高額な人頭税を課した。
  - グラナダにはアラブやアフリカで迫害を受けたムスリムやユダヤ人も集まってきて、経済的にも栄えた
  - グラナダの陥落は 1492 年。ナスル朝は宮殿を焼かずに鍵を渡して海へ去ったので、今もイスラムの繁栄を目にすることが出来る。
- もっともグラナダの人々の目には金持ちのアラブ人が去り、粗野で貧乏なスペイン人がやってきたと映ったようだ。

# 1　グラフィックスの準備

定規とコンパスは数学者の必携アイテムだが、ここで用いるのは幾何学とグラフィックス。最高水準の作品に迫るには十分な準備が要る

1. fvj4 with C.Reiter
   J の gl2 グラフィックスには C.Reiter の fvj4 を用いる。彼の著書「Fractal Visualization and J」の第 4 版では J8 の QT 版に対応済み。J の addon に入っている。
   グラフィックスのコマンドは gl2 を元に独自に構成されているが違和感はないし、原点が伝統的な左下なので扱いやすい
   （a）dwin.ijs NB. fvj4 の核
   （b）カンバスの準備と dwin の描画コマンド
   - dwin.ijs のロード　　load’addons/graphics/fvj4/dwin.ijs’
   - `setWIN\_WH 500 500 // setWIN_WH 1000 1000`　NB. 額縁の号数 (ピクセルベース）
   - `_1 _1 1 1 dwin ''` NB. 描く数値の範囲。絵は額縁まで拡大される
   - グラフィックスコマンド dpoly dline dpixel

   （c）Tool Hokusai 次の 2 本のファイルで構成される。(ロードする)

   ```
   hokusai_tool_improve2.ijs
   hokusai_tool_bezier.ijs
   ```

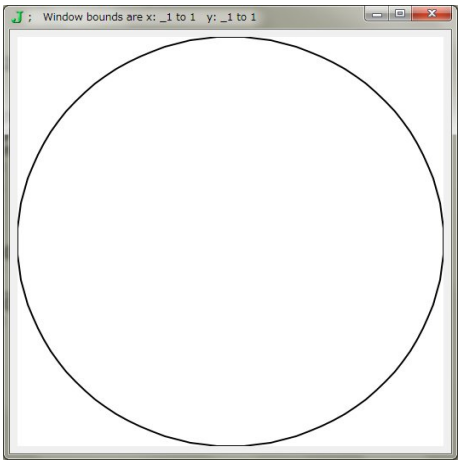
# 2　円のモザイクタイルを描く

1. 一個の円を描く。複素数とオイラーの有名な公式を用いる

   ```
   circ=: 3 : ' clean +. r. steps _1p1 1p1 100'
   NB. verb style
   ```

   steps　$-\pi$ から $\pi$ までを 100 ステップで刻む
   r.　$e^{i\theta} = \cos\theta + \sin i\theta$ を掛ける
   +.　複素数を分離して x , y の値を求める。gl2 には複素数の作図機能はない
   clean　numeric.ijs をロードすると入っている計算のゴミ取り

```
setWIN_WH 500 500
_1 _1 1 1 dwin ''
dline circ  ''
```




## 3 正多角形を描く

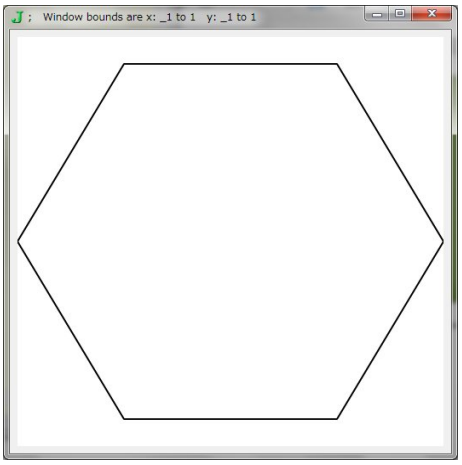
1. 正多角形の座標
   色々なツールを引っ張り出して、整理する。
   - 先ずは正多角形の各頂点の座標を求める。
   - (+.) で実虚部が分離でき、XY の座標となる
2. 簡潔な Script

```
poly0=: 3 : ' r. 2p1*(i.y)%y' NB. for dwin --> +.
poly=: [: clean +.@poly0
```

   2p1 * ((i.y)% y　$2\pi * \frac{012345}{5}$
   r.　オイラーの公式を適用
   +.　実数と虚数を分離し、XY の値とする
3. 計算と描画

   123 23 24 draw_poly_color poly 6

```
   poly 6
   1         0
 0.5  0.866025
_0.5  0.866025
  _1         0
_0.5 _0.866025
 0.5 _0.866025
```




4. カンバスをセットしてポリゴンを描く Script （モノクロとカラー版）

```
popup_win =: 3 : 0
```

```
NB. y is plot 6
setWIN_WH 500 500
(; 2# L:0 {@>find_maxmin tmp0) dwin ''
)

draw_poly=: 3 : 0
popup_win y
dpoly y
:
Color=. x
popup_win y
Color dpoly y
)
```

- 単項：両項型兼用
- 両項では x で RGB=カラーを指定する

## 4 斉時変換 (同時変換) を使いこなす

Homogenious Coordinates 斉時変換や同時変換と訳されるが数学の表現ではなく、グラフィックスの用語のようだ。

回転、拡大、移動を同時に行うため $3 \times 3$ のマトリクスに拡大する。

Script は C.Reiter を基にしている。

1. 回転

```
   clean  rotm 1r3p1
      0.5 0.866025 0
_0.866025      0.5 0
       0        0 1
```

2. 移動

```
   transm 2 3
1 0 0
0 1 0
2 3 1
```

3. 拡大

```
   elongm 2 4
2 0 0
```

```
0 4 0
0 0 1
```

4. mp mp=:  +/ . * ) 内積
5. 3 つを合成したマトリクス

```
   (rotm 1r3p1) mp (transm 2 3)  mp elongm 2 4
       1 3.4641 0
_1.73205      2 0
       4     12 1
```

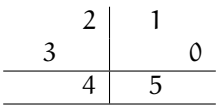
6. Script

```
NB. homogeneous Coordinates
NB. elongation(scaling) rotate transpose
elongm=: 3 :  '(y,1)* =i.3'
NB. elongm 2 3
rotm=: (cos, sin,0:),(-@sin,cos,0:),: 0: ,0:,1:
NB. rotm 1r4p1
transm=: 3 : '(=i.2), y,1'
NB. transm 2 3
```

## 4.1 6 角形の回転

1. 正 6 角形を描く

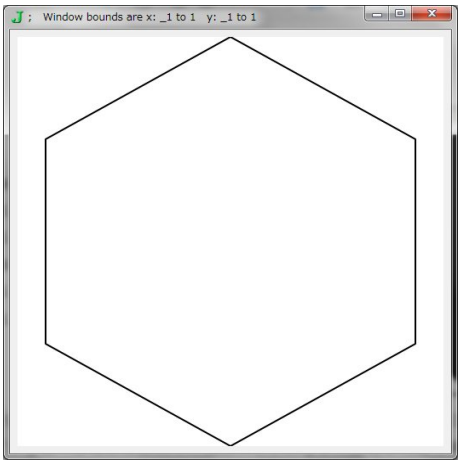
```
draw_poly poly 6
   poly 6
   1         0
 0.5  0.866025  NB. top right
_0.5  0.866025  NB. top left
  _1         0
_0.5 _0.866025  NB. bottom left
 0.5 _0.866025  NB. bottom right
  X      Y
```




2. 90 度回転すると、頂点が上に来る。この図（数値) を原図とする

```
a=.clean }:"1((poly 6),.1) mp rotm 1r2p1

        0    1  NB. top
_0.866025  0.5
_0.866025 _0.5
        0   _1 NB. bottom
 0.866025 _0.5
 0.866025  0.5
  X         Y
```




3. 斉時変換ではボトムを固定しない。しかし、回転でボトムを 0 にしないと漂流する羽目になる
4. しかしペダルを並べる場合は回転時に最初にボトムを 0 に固定した上で移動したほうが手数が格段に少なくなる

```
     0 1 + "1 a
        0   2
_0.866025 1.5
_0.866025 0.5
        0   0
 0.866025 0.5
 0.866025 1.5
```

## 4.2 transm とは

移動は斉時変換に位置を指定すれば出来る。これはマトリクスの内積での足し算である

1. XY=2 3 の移動と変換マトリクス

```
   transm 2 3
1 0 0
0 1 0
2 3 1
```

2. 斉時変換は内積演算

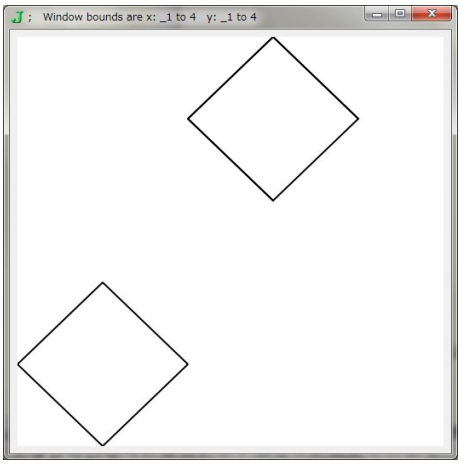
```
a=:(poly 4),.1      NB.4 角形の座標の右に 1の列を付加
b=: |: transm 2 3   NB. transpose
c=: a +/ . * b      NB. 内積
d=: 2 3 +"1 poly 4  NB. 列方向に (2 3) を加える
```

```
+--------+-----+-----+---+
| 1  0 1|1 0 2|3 3 1|3 3|
| 0  1 1|0 1 3|2 4 1|2 4|
|_1  0 1|0 0 1|1 3 1|1 3|
| 0 _1 1|     |2 2 1|2 2|
+-------+-----+-----+---+
  a       b     c     d
```

3. 単なる足し算でも同じに。

```
2 3 + "1 poly 4
```

4. graphics



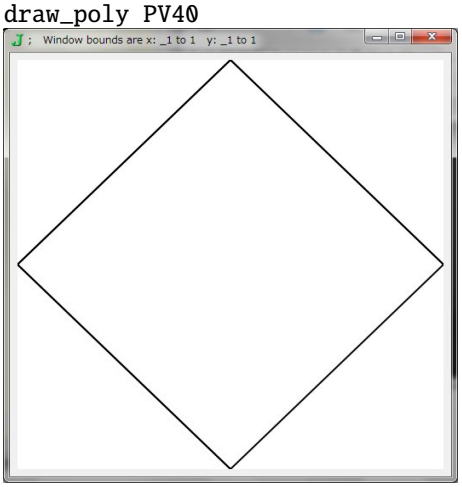

## 4.3 センターを入込む

1. 最初の正４角形。センターはこの図を３回回転させて１６角形を作っている

```
PV40=: clean }:"1 ((+. poly0 4),.1)
   PV40
 1  0  NB. NB. E
 0  1  NB. top=N
_1  0  NB. W
 0 _1  NB. bottom=S

  X         Y
```

draw_poly PV40



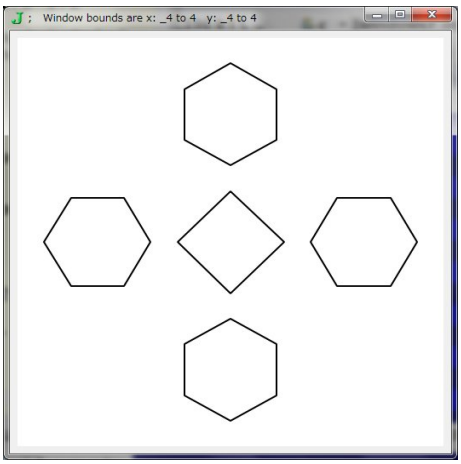

2. テストの NWSE の作図スクリプト

```
alh_V_fig0=: 3 : 0
tmp0=.PV40
tmp1=. test_parts_V1 ''
popup_win ;tmp1
dpoly L:0 tmp0
dpoly L:0 tmp1
NB. 100 100 100 dpoly L:0 tmp1
)
```

3. 回転角と正方形の座標を与えて斉時変換を行う。ボトムを 0 にしてあるので異動が簡単

```
test_parts_V1=: 3 : 0
NB. Usage: make_parts ''
tmp0=.clean }:"1 (PV60,.1) mp transm 0 1                    NB. N
tmp1=.clean }:"1 (PV60,.1) mp (rotm 1r2p1) mp transm _1 0  NB. W
tmp2=.clean }:"1 (PV60,.1) mp (rotm 1p1) mp transm  0 _1   NB. S
tmp3=.clean }:"1 (PV60,.1) mp (rotm _1r2p1) mp transm  1 0 NB. S
tmp0;tmp1;tmp2;tmp3
)
```

4. 成果は



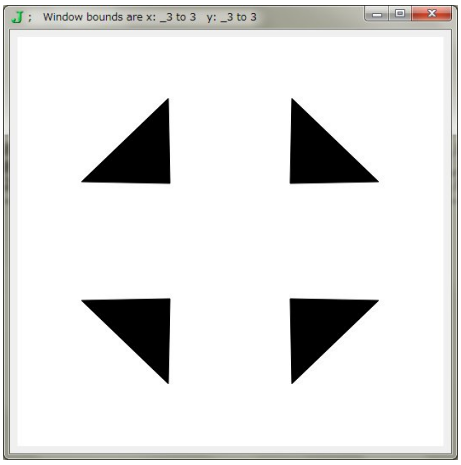

## 5　ペダルを差分で展開する

花弁は幾つかのパーツからなるボックスで構成された群になる。

これを差分指標で展開すると図形が散らばるか、一つずつ描いていたのでは手間なので、2 重ボックスで展開する

- 一群の花弁

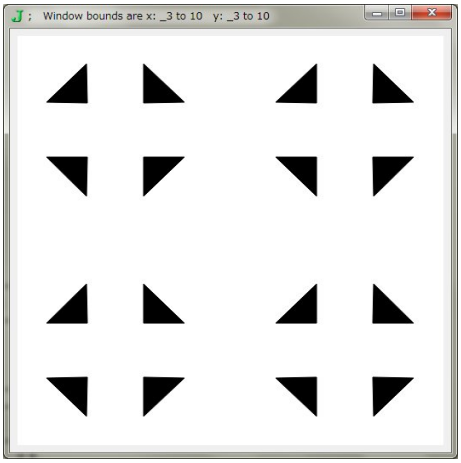
```
   mk_parts_No4_1 ''
+------------------+------------------+-----------------+-------------------+
|_0.848528 0.848528|0.848528 _0.848528|0.848528 0.848528|_0.848528 _0.848528|
|_0.872552   2.0973|0.872552   _2.0973|  2.0973 0.872552|  _2.0973 _0.872552|
|  _2.0973 0.872552|  2.0973 _0.872552|0.872552   2.0973|_0.872552   _2.0973|
+------------------+------------------+-----------------+-------------------+
```




- 2 重ボックスで花弁を差分位置に展開する

```
  ,. a1 calc_parts_box a
+--------------------------------------+
|+------------------+----------------+  |
||_0.848528 7.84853 |6.15147 7.84853 |  |
||_0.872552  9.0973 |6.12745  9.0973 |  |
||  _2.0973 7.87255 | 4.9027 7.87255 |  |
|+------------------+----------------+  |
||_0.848528 0.848528|6.15147 0.848528|  |
||_0.872552   2.0973|6.12745   2.0973|  |
||  _2.0973 0.872552| 4.9027 0.872552|  |
|+------------------+----------------+  |
+--------------------------------------+
|+------------------+-----------------+ |
||0.848528 6.15147  |7.84853 6.15147  | |
||0.872552  4.9027  |7.87255  4.9027  | |
||  2.0973 6.12745  | 9.0973 6.12745  | |
|+------------------+-----------------+ |
||0.848528 _0.848528|7.84853 _0.848528| |
||0.872552   _2.0973|7.87255   _2.0973| |
||  2.0973 _0.872552| 9.0973 _0.872552| |
|+------------------+-----------------+ |
+--------------------------------------+
|+-----------------+----------------+   |
||0.848528 7.84853 |7.84853 7.84853 |   |
||  2.0973 7.87255 | 9.0973 7.87255 |   |
||0.872552  9.0973 |7.87255  9.0973 |   |
```

```
|+-----------------+----------------+   |
||0.848528 0.848528|7.84853 0.848528|   |
||  2.0973 0.872552| 9.0973 0.872552|   |
||0.872552   2.0973|7.87255   2.0973|   |
|+-----------------+----------------+   |
+---------------------------------------+
|+-------------------+-----------------+|
||_0.848528 6.15147  |6.15147 6.15147  ||
||  _2.0973 6.12745  | 4.9027 6.12745  ||
||_0.872552  4.9027  |6.12745  4.9027  ||
|+-------------------+-----------------+|
||_0.848528 _0.848528|6.15147 _0.848528||
||  _2.0973 _0.872552| 4.9027 _0.872552||
||_0.872552   _2.0973|6.12745   _2.0973||
|+-------------------+-----------------+|
+---------------------------------------+
```




- 見た目よりも大幅な省力化がなされている
- スクリプト

```
   calc_parts_box=:4 : 0
NB. EXP calc_parts_box0 PARTS
 ANS=. <<''
for_ctr. i. # y do.
tmp=. x + "1 L:0 ctr{y
ANS=. ANS , <tmp
end.
 }.ANS
)
```

- 差分には基点は反映していなかったが、反映するものも作った

```
   N4PARAM2
+---------+---+---+
|10.5 10.5|7 0|0 7|
+---------+---+---+
基点     X 方向   Y 方向


   mk_diff_subXY=:4 : 0
NB. Usage: 4  5 mk_diff_sub0 MCPARAM //10 5;_2 5;6 _1
NB. x is size-of-matrix
NB. y is (<number of matrix),< base;diff x; diff y
'size_raw size_column'=. x
'base dfx dfy' =. y
X0=. |.{ base +"1 (|. i.size_column) */ dfx
tmp=. |: >{ L:0 X0 +"1 L:0 (|. i. size_raw) */ dfy
)
```